

# POKEBRO

## Autores:

- Miguel Ángel Abalde Paz
- Oscar Fernández Argiz
- Borja Díaz Vázquez
- Javier Pérez Vázquez

## Introducción de la aplicación

El siguiente manual de usuario replica el funcionamiento de nuestra aplicación, en este manual veremos tanto un sistema de recomendación de Pokemon como un sistema de valoración automática y valoración manual.

### Sistema de recomendación

El sistema de recomendación realizará varios procesos, entre ellos la tokenización de las descripciones de los Pokemon para su posterior creación del BagOfWords y de la matriz de distancias que serán usados para la recomendación de varios Pokemon a partir de una búsqueda inicial.

### Sistema de valoración

El sistema de valoración está compuesto por dos partes, la valoración manual que se basa en un formulario en el que el usuario puedes escribir sus valoraciones y la valoración automática en la que haremos uso de unos datos de entrenamiento para entrenar nuestro algoritmo de manera que podamos realizar un analisis de sentimientos de distintas opiniones sobre Pokemon. Para simular la existencia de multiples comentarios sobre Pokemon hemos generado un csv a partir de una búsqueda en twitter.

## ▼ SISTEMA DE RECOMENDACIÓN

El primer paso para crear nuestro sistema de recomendación será la importación de distintas herramientas que nos permitan empezar a trabajar.

Importamos **pandas** para trabajar con nuestros CSV a través de Dataframes, además de importar la herramienta **nltk** que nos ayudará en la tarea de tokenización de las descripciones.

```
import pandas as pd
from nltk.tokenize import word_tokenize
```

```

from nltk.corpus import stopwords
from nltk.stem import PorterStemmer

import nltk
nltk.download('punkt')
nltk.download('stopwords')

[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data] Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data] Package stopwords is already up-to-date!
True

```

El primer paso será el de cargar nuestro csv, en este caso **pokemon\_info.csv**. Este csv contiene a todos los Pokemon existentes además de estadísticas sobre ellos y una descripción de la Pokedex como se puede observar en la tabla.

```

rawData = pd.read_csv('pokemon_info.csv')
rawData

```

	id	identifier	height	weight	base_experience	order	is_default	flavor_text
0	1	bulbasaur	7	69	64	1	1	A strange plant on its back b
1	2	ivysaur	10	130	142	2	1	When the bulb on its back grows larger it appears
2	3	venusaur	20	1000	236	3	1	The plant blooms when it is absorbing solar energy
3	4	charmander	6	85	62	5	1	Obviously prefers hot places. When it rains, it seeks shelter.

El siguiente paso será la tokenización de las descripciones, para ello debemos recorrer nuestro dataframe. En cada iteración obtendremos la información de la columna *flavor\_text* de nuestro dataframe y le aplicaremos la tokenización fila por fila, de manera que el resultado será una nueva columna en el dataframe *processed\_text* que contendrá el texto procesado.

```

ps = PorterStemmer()
preText = []

i=0;

```

```
for row in rawData.itertuples():

    text = word_tokenize(rawData.iloc[i][7])
    stops = set(stopwords.words("english"))
    text = [ps.stem(w) for w in text if not w in stops and w.isalnum()]
    text = " ".join(text)

    preText.append(text)
    i=i+1
preData = rawData
preData['processed_text'] = preText

preData
```

	id	identifier	height	weight	base_experience	order	is_default	flavor_te
0	1	bulbasaur	7	69	64	1	1	A strange seed was planted on its back at birth.
1	2	ivysaur	10	130	142	2	1	When the bulb on its back grows large it appears.
2	3	venusaur	20	1000	236	3	1	The plant blooms when it is absorbing solar energy.
3	4	charmander	6	85	62	5	1	Obviously a fire-type, it likes to be in sunny places where it can bask.

Ejemplo de descripción sin tokenizar.

```
preData.iloc[0][7]

'A strange seed was planted on its back at birth. The plant sprouts and grows with this pokémon'
```

Ejemplo de descripción tokenizada.

```
preData.iloc[0][8]

'A strang seed plant back birth the plant sprout grow pokémon'
```

Una vez tokenizadas las descripciones creamos el *BagOfWords*. Para la creación del *BagOfWords* importamos la librería **TfidfVectorizer** que contendrá las herramientas necesarias para su creación.

El *BagOfWords* contendrá las palabras contenidas en las descripciones tokenizadas y su frecuencia de aparición en todas ellas. Para ello creamos el *BagOfWords* llamando a `TfidfVectorizer()` y posteriormente hacemos el fit sobre la columna de texto procesado y realizamos el transform sobre la misma columna lo que nos resulta en nuestra variable `textsBoW` que contiene lo descrito anteriormente

```
from sklearn.feature_extraction.text import TfidfVectorizer
```

```
bagOfWordsModel = TfidfVectorizer()
bagOfWordsModel.fit(preData['processed_text'])
textsBoW= bagOfWordsModel.transform(preData['processed_text'])
print("Finished")
```

```
Finished
```

```
textsBoW.shape
```

```
(718, 3017)
```

Mostramos el contenido de nuestra variable **textsBoW**

```
print(textsBoW)
```

```
(0, 2630)    0.1813053884156978
(0, 2499)    0.3573407812365763
(0, 2457)    0.3573407812365763
(0, 2236)    0.3573407812365763
(0, 1910)    0.12021381063868418
(0, 1883)    0.5713665604880965
(0, 1165)    0.25554053485302297
(0, 266)     0.34147466227401224
(0, 199)     0.24745512257122929
(1, 2921)    0.21617797692488688
(1, 2463)    0.3095255010078092
(1, 1483)    0.30289507729672716
(1, 1440)    0.3170424554392978
(1, 1415)    0.30289507729672716
(1, 1230)    0.38756491433490514
(1, 1165)    0.26083173549586963
(1, 348)     0.38756491433490514
(1, 199)     0.2525789073530748
(1, 124)     0.23702762759148438
(1, 19)      0.2915986140166228
(2, 2630)    0.18617906553178423
(2, 2542)    0.3506538557982136
(2, 2471)    0.3113980855493526
(2, 2387)    0.38990962604707463
(2, 2238)    0.36694647251456575
:           :
(716, 2941)  0.21033961302520698
(716, 2932)  0.2825961560092591
```

```

(716, 2921)    0.18182748490483877
(716, 2583)    0.18782899352928542
(716, 2452)    0.3259811874146669
(716, 2073)    0.3259811874146669
(716, 1910)    0.10320555623244386
(716, 1471)    0.17357210600185285
(716, 1453)    0.27396349588051405
(716, 1442)    0.2825961560092591
(716, 1133)    0.2666646857399908
(716, 1047)    0.2547653136473603
(716, 959)     0.2931616781136678
(716, 586)     0.3259811874146669
(716, 23)      0.2666646857399908
(717, 2921)    0.20815044980988684
(717, 2232)    0.30526943885431296
(717, 2125)    0.3731731252040117
(717, 2082)    0.3235072905433278
(717, 1944)    0.19255717821146212
(717, 1380)    0.3731731252040117
(717, 945)     0.3356023716563947
(717, 790)     0.3731731252040117
(717, 683)     0.3731731252040117
(717, 124)     0.2282258720446837

```

Mostramos las palabras contenidas en nuestro BagOfWords.

```
bagOfWordsModel.get_feature_names()
```

```

'fact',
'factori',
'fahrenheit',
'faiblit',
'fail',
'faint',
'faintest',
'fair',
'fairi',
'faisant',
'fait',
'fall',

'fallen',
'fan',
'fang',
'far',
'fascin',
'fast',
'faster',
'fatal',
'fatigu',
'faufil',
'favorit',
'façon',
'fear',
'feather',
'feebe',
'feed',
'feel',
'feeler',
'feet',

```

```
'fell',
'femal',
'femel',
'fend',
'fer',
'ferait',
'fermit',
'feroci',
'ferri',
'festiv',
'feu',
'feud',
'feuill',
'fever',
'fickl',
'field',
'fierc',
'fight',
'fil',
'fill',
'film',
'filthi',
'fin',
'find',
'fine',
'finger',
'finish',
'finiss',
'finiss'
```

Una vez creado el BagOfWords el siguiente paso será crear la matriz de distancias. Para su creación importaremos **pairwise\_distances**.

La matriz de distancia usará la métrica *manhattan* para determinar la distancia entre los elementos. Al realizar el calculo de distancias entre los elementos lo que hace esta librería con la metrica utilizada es crear una matriz que contendrá todos los Pokemon de nuestro csv y además estos estarán colocados entre ellos según su parecido.

```
from sklearn.metrics import pairwise_distances

distance_matrix= pairwise_distances(textsBoW,textsBoW ,metric='manhattan')

print(distance_matrix.shape)

(718, 718)
```

El siguiente paso será indicarle un valor de busqueda inicial en este caso Raikou. Además mostramos su posición en el csv, en nuestro caso **242**

```
searchTitle = "raikou"
indexOfTitle = preData[preData['identifier']==searchTitle].index.values[0]
indexOfTitle
```

242

Una vez hemos introducido el elemento de búsqueda inicial crearemos la variable *distance\_scores* que contendrá la puntuación de cada elemento con respecto a nuestro valor de búsqueda inicial *list(enumerate(distance\_matrix[indexOfTitle]))*. Cada uno de los elementos puede tener una puntuación del 0 al 10 debido a la métrica Manhattan, siendo 0 lo mas parecido y 10 lo mas alejado. Como podemos observar al mostrar *distance\_scores* el elemento 242 tiene una puntuación de 0 ya que es el propio elemento que se ha usado como búsqueda inicial.

```
distance_scores = list(enumerate(distance_matrix[indexOfTitle]))
distance_scores
```

```
(654, 6.493329716277952),
(655, 6.2788553198725205),
(656, 6.4929019130862695),
(657, 6.534950311603171),
(658, 5.444202702390927),
(659, 6.256966887487028),
(660, 6.345254995759625),
(661, 6.3694694266632),
(662, 6.3952592762680975),
(663, 6.4741487898671926),
(664, 7.127544060420643),
(665, 6.333261623350927),
(666, 5.967124282046249),
(667, 4.980078553055622),
(668, 6.128645855031428),
(669, 6.379810550637847),
(670, 6.544954550892636),
(671, 6.144056255584562),
(672, 6.364783433526465),
(673, 6.494794013741004),
(674, 6.768479042431803),
(675, 6.237096937477309),
(676, 6.0163258382507125),
(677, 6.522285494943288),
(678, 6.738987468777705),
(679, 6.939677828742625),
(680, 6.336480447444584),
(681, 6.304749434379634),
(682, 6.69132055153162),
(683, 6.556400563827474),
(684, 6.814521712614525),
(685, 6.208392399731808),
(686, 6.253752263918647),
(687, 6.254641675645349),
(688, 5.541540227676474),
(689, 6.3959347812609),
(690, 6.701669523507811),
(691, 5.6972153587756065),
(692, 6.533074461342923),
(693, 6.037411062750829),
(694, 6.054582405804635),
(695, 6.612384206413666),
(696, 6.851195471659563),
(697, 6.452041159144723),
(698, 6.2997944179420795),
```

```
(699, 6.002563676046578),
(700, 6.8057483475678024),
(701, 6.796219175576346),
(702, 5.730545112124161),
(703, 5.731081975614449),
(704, 6.646537222628497),
(705, 6.209463118745262),
(706, 5.74735240251239),
(707, 6.318887058455744),
(708, 6.2311873136428835),
(709, 6.462161759060014),
(710, 6.858325333329546),
(711, 6.636153725100842),
(712, 6.357607615195945),
```

Una vez obtenidas las puntuaciones debemos ordenarlas para mostrar los 10 resultados mas parecidos. Obviamente hemos de omitir el primer elemento ya que es el propio elemento de la busqueda.

```
ordered_scores = sorted(distance_scores, key=lambda x: x[1])
ordered_scores
```

```
(622, 7.371676423757282),
(515, 7.373549553514724),
(614, 7.37487345519784),
(583, 7.376973039960314),
(504, 7.381202214587943),
(283, 7.3923875241745005),
(545, 7.394353132201952),
(636, 7.397344001874439),
(497, 7.398953752563901),
(306, 7.4059925129674475),
(535, 7.407023530417585),
(544, 7.407650802012088),
(327, 7.40893969690982),
(316, 7.409458399067618),
(336, 7.410085103647226),
(597, 7.412671699152224),
(337, 7.416970878659504),
(556, 7.417763496001415),
(601, 7.419633060032511),
(312, 7.4200830045852495),
(633, 7.421058548563986),
(627, 7.4244366458187985),
(588, 7.434075107081659),
(508, 7.436587423676933),
(261, 7.445540542559732),
(372, 7.446583501682547),
(368, 7.454698779211073),
(613, 7.458555390826619),
(511, 7.463075859114688),
(291, 7.4688572177190355),
(635, 7.4740500842565964),
(366, 7.478239510600754),
(611, 7.485109293733569),
(323, 7.489917403522872),
(512, 7.490180754286065),
(514, 7.495505084497085),
(528, 7.4960741713536105),
```



```
(345, 7.501895147629002),
(263, 7.505234647642963),
(568, 7.50532157399612),
(303, 7.510652869930076),
(292, 7.522676704594053),
(610, 7.525155216735169),
(565, 7.534005907542625),
(502, 7.535852083612483),
(498, 7.5485304683574475),
(634, 7.569389701954944),
(318, 7.598546775794969),
(591, 7.606133660383478),
(325, 7.612691020151545),
(620, 7.620165263566025),
(310, 7.636456971522124),
(276, 7.651667999718689),
(360, 7.663118567703742),

(287, 7.683565159298475),
(269, 7.709667245703216),
(260, 7.723563638830581),
(275, 7.727005264844204),
(286, 7.73656421660322),
(220, 7.7805710546417)
```

Cogemos los 11 primeros elementos mas parecidos a Raikou y omitimos el primero como dijimos anteriormente.

```
top_scores = ordered_scores[1:11]
top_scores
```

```
[(175, 4.955756377066811),
 (667, 4.980078553055622),
 (144, 5.094129296090852),
 (177, 5.154889759281152),
 (440, 5.216210756155929),
 (106, 5.268159255814101),
 (181, 5.314651096121145),
 (658, 5.444202702390927),
 (688, 5.541540227676474),
 (134, 5.551591859444804)]
```

```
top_indexes = [i[0] for i in top_scores]
top_indexes
```

```
[175, 667, 144, 177, 440, 106, 181, 658, 688, 134]
```

Por ultimo accedemos a predata con los identificadores obtenidos en los top\_indexes y mostramos el nombre correspondiente al elemento.

En este caso el resultado de recomendación con nuestro parametro de busqueda *Raikou* es el siguiente:

```
preData['identifier'].iloc[top_indexes]
```

```

175      togetic
667      pyroar
144      zapdos
177      xatu
440      chatot
106      hitmonchan
181      bellossom
658      bunnelby
688      barbaracle
134      jolteon
Name: identifier, dtype: object

```

## ▼ SISTEMA DE VALORACIÓN

### Valoración Automática

Una vez hemos descrito nuestro sistema de recomendación daremos paso a describir nuestro sistema de valoración.

En este sistema se importarán varias utilidades como en el sistema anterior.

Importamos pandas para trabajar con nuestros dataframes además de importar nuevamente la librería nltk para la tokenización de las palabras.

Ademas cabe destacar el import de **train\_test\_split** que nos permitirá dividir un dataset en bloques para posteriormente usarlo como entrenamiento de nuestro algoritmo de valoración.

```

import numpy as np
import pandas as pd

from subprocess import check_output

import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split

import nltk
from nltk.corpus import stopwords
from nltk.classify import SklearnClassifier

from subprocess import check_output

```

El siguiente paso como en el sistema de recomendación será la carga de nuestro CSV. En este caso *Sentiment.csv* es un csv que contiene distintas opiniones sacadas de Twitter con una columna de intención "Neutral", "Positive" o "Negative" que indica la intención del comentario. Este csv será utilizado para el entrenamiento de nuestro algoritmo de valoración automática.

```

data = pd.read_csv('Sentiment.csv')
data = data.head(13871)

```

data

	id	candidate	candidate_confidence	relevant_yn	relevant_yn_confidence
0	1	No candidate mentioned	1.0000	yes	1.0000
1	2	Scott Walker	1.0000	yes	1.0000
2	3	No candidate mentioned	1.0000	yes	1.0000
3	4	No candidate mentioned	1.0000	yes	1.0000
4	5	Donald Trump	1.0000	yes	1.0000
...	...	...	...	...	...
13866	13867	No candidate mentioned	1.0000	yes	1.0000
13867	13868	Mike Huckabee	0.9611	yes	1.0000
13868	13869	Ted Cruz	1.0000	yes	1.0000
13869	13870	Donald Trump	1.0000	yes	1.0000
13870	13871	Ted Cruz	0.9242	yes	0.9614

13871 rows × 21 columns

Antes de empezar a trabajar con el csv debemos darle el formato que queremos para adecuarlo a nuestras necesidades. En este caso cambiaremos el valor de la columna *Sentiment* sustituyendo los valores de texto por valores numéricos.

La transformación será la siguiente:

- Positive = 1
- Neutral = 0

- Negative = -1

```
data = data[['text', 'sentiment']]
data.loc[data.sentiment == 'Neutral', 'sentiment'] = 0
data.loc[data.sentiment == 'Positive', 'sentiment'] = 1
data.loc[data.sentiment == 'Negative', 'sentiment'] = -1
data
```

/usr/local/lib/python3.6/dist-packages/pandas/core/indexing.py:670: SettingWithCopyWarning: A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: <https://pandas.pydata.org/pandas-docs/stable/10min.html#copy-on-write>  
iloc.\_setitem\_with\_indexer(indexer, value)

/usr/local/lib/python3.6/dist-packages/ipykernel\_launcher.py:2: SettingWithCopyWarning: A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: <https://pandas.pydata.org/pandas-docs/stable/10min.html#copy-on-write>

/usr/local/lib/python3.6/dist-packages/ipykernel\_launcher.py:3: SettingWithCopyWarning: A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: <https://pandas.pydata.org/pandas-docs/stable/10min.html#copy-on-write>

This is separate from the ipykernel package so we can avoid doing imports until  
/usr/local/lib/python3.6/dist-packages/ipykernel\_launcher.py:4: SettingWithCopyWarning: A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: <https://pandas.pydata.org/pandas-docs/stable/10min.html#copy-on-write>  
after removing the cwd from sys.path.

	text	sentiment
0	RT @NancyLeeGrahm: How did everyone feel about...	0
1	RT @ScottWalker: Didn't catch the full #GOPdeb...	1
2	RT @TJMShow: No mention of Tamir Rice and the ...	0
3	RT @RobGeorge: That Carly Fiorina is trending ...	1
4	RT @DanScavino: #GOPDebate w/ @realDonaldTrump...	1
...	...	...
13866	RT @cappy_yarbrough: Love to see men who will ...	-1
13867	RT @georgehenryw: Who thought Huckabee exceede...	1
13868	RT @Lrihendry: #TedCruz As President, I will a...	1
13869	RT @JRehling: #GOPDebate Donald Trump says tha...	-1
13870	RT @Lrihendry: #TedCruz headed into the Presid...	1

13871 rows × 2 columns

Mostramos la frecuencia de aparición de cada valor numérico.

```
data['sentiment'].value_counts()
```

```
-1    8493
0     3142
1     2236
Name: sentiment, dtype: int64
```

El siguiente paso será el de tokenizar los comentarios del csv para posteriormente poder entrenar nuestro algoritmo. El proceso de tokenización es muy parecido al explicado anteriormente, recorreremos nuestro dataframe y realizaremos la tokenización de la columna con índice 1 que corresponde a los comentarios. Su tokenización usará como idioma de referencia el inglés y añadirá una nueva columna a nuestro dataframe que será *processed\_text*

```
ps = PorterStemmer()

preprocessedText = []

for row in data.itertuples():

    text = word_tokenize(row[1])
    ## Remove stop words
    stops = set(stopwords.words("english"))
    text = [ps.stem(w) for w in text if not w in stops and w.isalnum()]
    text = " ".join(text)

    preprocessedText.append(text)

preprocessedData = data
preprocessedData['processed_text'] = preprocessedText

preprocessedData
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:17: SettingWithCopyWarn
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

See the caveats in the documentation: <https://pandas.pydata.org/pandas-docs/stable/u>

text	sentiment	processed text
------	-----------	----------------

El siguiente paso será la creación del BagOfWords de la misma manera que lo hemos creado en el sistema de recomendación.

text	sentiment	processed text
RT @ScottWalker: Didn't catch the full		RT scottwalk did catch full gopdeb

```
from sklearn.feature_extraction.text import TfidfVectorizer
```

```
bagOfWordsModel = TfidfVectorizer()
bagOfWordsModel.fit(preprocessedData['processed_text'])
textsBow= bagOfWordsModel.transform(preprocessedData['processed_text'])
print("Finished")
```

Finished

```
textsBow.shape
```

```
(13871, 11723)
```

text	sentiment	processed text
RT @georgehenrw: Who thought		RT georgehenrw who thought

Una vez creado el BagOfWords daremos paso al entrenamiento de nuestro algoritmo. Para ello definimos nuestro algoritmo clasificador SVM con el parametro **kernel=linear** que supone más flexibilidad en la asignación de penalizaciones y funciones de coste.

Para el entrenamiento de nuestro algoritmo debemos definir 2 variables que son:

- X\_Train: Contiene una lista de valoraciones.
- Y\_Train: Contiene el sentimiento de cada una de las valoraciones, es decir positivo, neutro o negativo.

Estas dos variables se le pasarán al algoritmo con fit y este asociará ciertos patrones para el analisis de sentimientos de una frase o comentario.

```
from sklearn import svm
svc = svm.SVC(kernel='linear')
```

```
X_train = textsBow #Documentos
Y_train = data['sentiment']
Y_train=Y_train.astype('int')
svc.fit(X_train, Y_train)
```

```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='linear',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

Cargamos el csv que usaremos para testear nuestro algoritmo de valoración automática. Este csv contendrá una serie de comentarios de los cual nuestro algoritmo deberá predecir su sentimiento

```
testData = pd.read_csv('semeval-2017-test.csv', delimiter=' ')
testData = testData.head(1000)
testData
```

	label	text
0	0	Trump is building a wall on the Mexican border...
1	-1	@lasinferencias & the WALL Trump wants to buil...
2	-1	President Elect? More like President Erect! A ...
3	0	Ok, I know a lot of you think a wall on the Me...
4	0	The Great Mexican Wall Deception: Trump's Amer...
...	...	...
995	-1	Listening to Melania practice the speech Miche...
996	-1	is there anyone worse than Grayson Allen
997	1	Grayson Allen is good. I think he can pay next...
998	1	We need to start talking more about how Grayso...
999	1	I never thought I would say this but Grayson A...

1000 rows × 2 columns

En este caso tambien debemos realizar la tokenización del texto para posteriormente pasarselas a nuestro algoritmo. El proceso es el mismo que para los casos anteriores.

```
ps = PorterStemmer()

preprocessedText = []

for row in testData.itertuples():

    text = word_tokenize(row[2])
    ## Remove stop words
    stops = set(stopwords.words("english"))
    text = [ps.stem(w) for w in text if not w in stops and w.isalnum()]
    text = " ".join(text)

    preprocessedText.append(text)

preprocessedDataTest = testData
preprocessedDataTest['processed_text'] = preprocessedText

preprocessedDataTest
```

	label	text	processed_text
0	0	Trump is building a wall on the Mexican border...	trump build wall mexican border stop herrion c...
1	-1	@lasinferencias & the WALL Trump wants to buil...	lasinferencia wall trump want build I research...
2	-1	President Elect? More like President Erect! A ...	presid elect more like presid erect A wall On ...
3	0	Ok, I know a lot of you think a wall on the Me...	Ok I know lot think wall mexican border insan ...
4	0	The Great Mexican Wall Deception: Trump's Amer...	the great mexican wall decept trump america al...
...	...	...	...
995	-1	Listening to Melania practice the speech Miche...	listen melania practic speech michel obama wro...
996	-1	is there anyone worse than Grayson Allen	anyon wors grayson allen
997	1	Grayson Allen is good. I think he can pay ...	grayson allen good I think pay next level ...

Mostramos el numero de valores negativos, positivos y neutros

```
testData['label'].value_counts()
```

```
0      474
-1     341
1      185
Name: label, dtype: int64
```

Creamos la variable textsBowTest con el transform del BagOfWords sobre la columna del texto procesado.

```
textsBowTest= bagOfWordsModel.transform(preprocessedDataTest['processed_text'])
print("Finished")
```

```
Finished
```

```
textsBowTest.shape
```

```
(1000, 11723)
```

Una vez obtenemos el textsBowTest lo igualamos a la variable X\_test que le pasaremos al algoritmo para que realice la predicción sobre las descripciones.

```
X_test = textsBowTest
```

```
predictions = svc.predict(X_test)
```



```
print("Finished")
```

Finished

Mostramos las predicciones obtenidas:

```
print(predictions)
```

```
[ -1 -1 -1 -1 -1 -1 -1 -1 -1 -1  0 -1  0 -1 -1 -1 -1 -1  0  1 -1 -1 -1
  -1 -1 -1 -1 -1 -1 -1 -1 -1 -1  0 -1  0 -1  0  1  0 -1  0 -1 -1 -1  1 -1 -1
  -1 -1 -1  0 -1 -1 -1 -1  0 -1 -1 -1  0 -1 -1 -1 -1  0  0  0  0  0 -1
  -1  0 -1  0  0  0  0 -1 -1  0 -1  0  0 -1 -1 -1  0 -1  0 -1  0  0 -1 -1
  -1 -1 -1 -1 -1 -1  0  0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1  0  0  0
  -1 -1 -1 -1 -1 -1  0  0  0 -1  0 -1 -1 -1 -1 -1 -1 -1 -1  1 -1 -1 -1
  0 -1 -1 -1  0 -1 -1 -1 -1  0  0  1 -1  0 -1 -1 -1 -1  1 -1  0 -1 -1 -1
  -1  1 -1 -1  0 -1 -1  0  0  0 -1 -1  0  0 -1 -1 -1 -1  1 -1  0  1 -1
  -1 -1 -1 -1  0  0 -1 -1 -1  0 -1 -1 -1 -1  0  0 -1 -1 -1 -1  0 -1 -1 -1
  -1 -1 -1 -1 -1  0 -1 -1  0  0 -1 -1  0  1  0  0  0 -1 -1 -1  0 -1  0  0
  0 -1 -1 -1 -1 -1 -1 -1 -1 -1  0 -1 -1 -1  0 -1 -1 -1 -1 -1  0 -1 -1
  -1 -1 -1  0  0  0 -1 -1 -1  0 -1 -1  0 -1 -1 -1 -1  1  0 -1  0 -1 -1
  -1 -1  0 -1 -1 -1 -1 -1 -1  0 -1 -1  0 -1 -1 -1 -1  0  0 -1 -1 -1 -1
  -1  0  0  1 -1 -1 -1  0 -1 -1 -1 -1 -1 -1  0 -1 -1  0  0  0  0 -1 -1
  0  0  1 -1  0 -1 -1  0 -1  0  0  0 -1 -1 -1 -1 -1 -1 -1 -1 -1  0 -1
  0  0  0 -1 -1 -1  0 -1 -1  0  0 -1 -1 -1 -1 -1  0  0  0 -1 -1 -1 -1  0
  -1  0  0  0  0 -1  0 -1  0 -1 -1 -1 -1  1 -1 -1 -1 -1 -1  0  0 -1  1 -1
  0 -1 -1 -1  0 -1 -1 -1 -1  1 -1 -1 -1  0 -1  0 -1 -1 -1  0 -1 -1 -1
  -1 -1  0 -1 -1 -1 -1 -1  0 -1  0  0 -1 -1 -1 -1 -1  0 -1 -1 -1 -1  0
  0  0 -1  0 -1  0  0 -1 -1  0  0  0 -1  0 -1 -1  0  0 -1  0  0 -1 -1  0
  -1 -1 -1 -1 -1 -1  0 -1  1  0  0 -1  1 -1 -1  0  0  0  0  0 -1 -1 -1 -1
  0 -1 -1  0 -1  0 -1  0  0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1  1 -1
  -1  0  0 -1 -1 -1 -1 -1 -1 -1  0 -1  0 -1 -1 -1 -1  0 -1 -1 -1 -1 -1
  -1 -1 -1 -1  0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1  1 -1 -1  0
  -1 -1  0 -1 -1 -1  0  0 -1 -1 -1 -1 -1 -1 -1  0 -1 -1  0 -1 -1  0 -1
  -1 -1 -1  0 -1 -1 -1 -1 -1  0 -1 -1 -1 -1 -1 -1 -1  0 -1 -1 -1 -1 -1
  -1  0 -1 -1 -1 -1 -1 -1 -1 -1  0 -1 -1 -1  0 -1 -1 -1 -1  0 -1  0 -1
  -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1  0 -1 -1
  0  0 -1 -1 -1 -1 -1  0  0 -1 -1 -1  0  0 -1 -1  0 -1 -1 -1 -1  0 -1 -1
  -1 -1 -1 -1 -1 -1  0  0  0 -1 -1 -1  0  0 -1  1 -1 -1 -1 -1  0 -1  0 -1
  -1  0  0 -1 -1  0 -1  0 -1  0 -1 -1 -1  0 -1 -1  0  0 -1 -1 -1  0 -1  1
  0 -1  0 -1  0  0  0 -1 -1 -1  1  0 -1 -1 -1 -1 -1 -1 -1  1 -1 -1 -1
  -1 -1 -1 -1 -1 -1  0 -1 -1 -1 -1 -1 -1 -1 -1 -1  0 -1 -1  0 -1 -1 -1
  -1  0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1  0 -1 -1 -1 -1 -1
  -1 -1 -1 -1 -1  0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1  1 -1 -1 -1 -1  0 -1
  -1 -1 -1 -1 -1  0  0 -1 -1 -1 -1  0 -1 -1 -1 -1 -1  0 -1 -1  0 -1 -1
  -1 -1 -1 -1 -1 -1  0 -1 -1 -1 -1  0  0 -1 -1 -1  0 -1 -1 -1 -1  0  1
  -1  0 -1 -1 -1 -1 -1 -1 -1 -1 -1  0  0 -1 -1 -1 -1  0 -1  0 -1 -1 -1
  0 -1 -1 -1 -1  0 -1 -1 -1  0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1  0 -1 -1
  -1 -1  0 -1 -1 -1 -1 -1 -1 -1 -1 -1  0 -1 -1 -1 -1 -1 -1  0 -1 -1 -1
  -1 -1 -1 -1 -1 -1 -1 -1  0 -1  0 -1 -1  0  0  0 -1 -1 -1 -1 -1 -1 -1
  1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1  0 -1]
```

El siguiente paso será el de valorar la predicción que ha realizado nuestro algoritmo para ello usaremos la herramienta **classification\_report**, esta herramienta nos permite obtener unas estadísticas acerca de la predicción realizada. Para ello debemos llamar a `classification_report`

con los argumentos `Y_test`, que corresponde a la verdadera intención de las descripciones analizadas, y `predictions` que contendrá las predicciones realizadas.

Como podemos observar el ratio de acierto es bastante bajo, esto se debe a que nuestros datos de entrenamiento eran bastante escasos. Si aumentamos el numero de datos de

```
from sklearn.metrics import classification_report

Y_test = testData['label']

print (classification_report(Y_test, predictions))
```

	precision	recall	f1-score	support
-1	0.41	0.88	0.56	341
0	0.57	0.29	0.39	474
1	0.52	0.08	0.13	185
accuracy			0.45	1000
macro avg	0.50	0.42	0.36	1000
weighted avg	0.51	0.45	0.40	1000

Por último y como pequeña simulación de nuestro sistema en concreto realizaremos otra predicción pero esta vez sobre un csv de opiniones de Pokemon obtenido de mensajes de Twitter. El procedimiento a seguir para su predicción es el mismo que el seguido por el csv anterior por lo que no entraremos en detalles.

Carga del csv.

```
pokemonDataComments = pd.read_csv('PokeInfoDef.csv')
pokemonDataComments["text"] = pokemonDataComments["text"].map(str)
pokemonDataComments
```

	tweetDate	twitterId	handle	text	
0	Sat Jan 23 16:39:38 +0000 2021	934948599028502528	PeaceOfYoshi	Meganium best... cause, well ive explained it ...	https://twit
1	Fri Jan 22 00:01:19 +0000 2021	43285703	docvalentine	@nosplendorr hmmm based on his pokemon opinion...	https://tv
2	Sat Jan 23 16:56:15 +0000 2021	714849904888299521	StealthHawkDude	@PeaceOfYoshi i see we differ in a lot of poke...	https://twitter.c
3	Sun Jan 24 08:35:16 +0000 2021	193929402	KaggyFilms	Thats gonna be a yikes from me Cotton.	https://t
4	Sat Jan 23 07:24:07 +0000 2021	929101255284285440	ItsAbouTimeJoey	Posted a Pokémon tier list. \n\nGonna lose som...	https://twitter.
...	...	...	...	...	...
576	Wed Oct 16 14:25:32 +0000 2019	719461958550822912	zane-flynt	imagine having real Pokémon opinions.. couldn't...	https
577	Fri Jul 14 16:46:29 +0000 2017	2267742470	Poijz	I liked a @YouTube video https://t.co/T31Tljz1...	t
...	...	...	...	...	...

Tokenización de los comentarios.

```

+00000                                couldn'

psOpinions = PorterStemmer()

preprocessedPokeOpinions = []

for row in pokemonDataComments.itertuples():
    textOpinions = word_tokenize(row[4])
    stopsOpinions = set(stopwords.words("english"))
    textOpinions = [psOpinions.stem(w) for w in textOpinions if not w in stopsOpinions and
    textOpinions = " ".join(textOpinions)

    preprocessedPokeOpinions.append(textOpinions)

preprocessedDataOpinions = pokemonDataComments
preprocessedDataOpinions['processed_text'] = preprocessedPokeOpinions

```

preprocessedDataOpinions

**tweetDate**                      **twitterId**                      **handle**                      **text**

Creación del bag of words y de la variable textsBoWOpinions.

```

n      10:39:30      021018500078507528      PeaceOfYoshi      -      cause, well live      https://twit
bagOfWordsModelOpinions = TfidfVectorizer()
bagOfWordsModelOpinions.fit(preprocessedDataOpinions['processed_text'])
textsBowOpinions= bagOfWordsModelOpinions.transform(preprocessedDataOpinions['processed_text'])
print("Finished")

```

Finished

Sat Jan 23

textsBowOpinions.shape

(581, 1885)

Sun Jan

Predicción realizada:

10000  
10000

10000

X\_testOpinions = textsBowOpinions

predictions = svc.predict(X\_test)

print(predictions)

```

[-1 -1 -1 -1 -1 -1 -1 -1 -1 -1  0 -1  0 -1 -1 -1 -1 -1  0  1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1  0 -1  0 -1  0 -1  0 -1 -1 -1  1 -1 -1
-1 -1 -1  0 -1 -1 -1 -1  0 -1 -1 -1  0 -1 -1 -1 -1  0  0  0  0  0 -1
-1  0 -1  0  0  0  0 -1 -1  0 -1  0  0 -1 -1 -1  0 -1  0 -1  0  0 -1 -1
-1 -1 -1 -1 -1 -1  0  0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1  0  0  0
-1 -1 -1 -1 -1 -1  0  0  0 -1  0 -1 -1 -1 -1 -1 -1 -1 -1  1 -1 -1 -1
 0 -1 -1 -1  0 -1 -1 -1 -1  0  0  1 -1  0 -1 -1 -1 -1  1 -1  0 -1 -1 -1
-1  1 -1 -1  0 -1 -1  0  0  0 -1 -1  0  0 -1 -1 -1 -1 -1  1 -1  0  1 -1
-1 -1 -1 -1  0  0 -1 -1 -1  0 -1 -1 -1 -1  0  0 -1 -1 -1 -1  0 -1 -1 -1
-1 -1 -1 -1 -1  0 -1 -1  0  0 -1 -1  0  1  0  0  0 -1 -1 -1  0 -1  0  0
 0 -1 -1 -1 -1 -1 -1 -1 -1 -1  0 -1 -1 -1  0 -1 -1 -1 -1 -1 -1  0 -1 -1
-1 -1 -1  0  0  0 -1 -1 -1  0 -1 -1  0 -1 -1 -1 -1  1  0 -1  0 -1 -1
-1 -1  0 -1 -1 -1 -1 -1 -1 -1  0 -1 -1  0 -1 -1 -1 -1  0  0 -1 -1 -1
-1  0  0  1 -1 -1 -1  0 -1 -1 -1 -1 -1 -1  0 -1 -1  0  0  0  0  0 -1 -1
 0  0  1 -1  0 -1 -1  0 -1  0  0  0 -1 -1 -1 -1 -1 -1 -1 -1 -1  0 -1
 0  0  0 -1 -1 -1  0 -1 -1  0  0 -1 -1 -1 -1 -1  0  0  0 -1 -1 -1 -1  0
-1  0  0  0  0 -1  0 -1  0 -1 -1 -1 -1  1 -1 -1 -1 -1 -1  0  0 -1  1 -1
 0 -1 -1 -1  0 -1 -1 -1 -1 -1  1 -1 -1 -1  0 -1  0 -1 -1 -1  0 -1 -1 -1
-1 -1  0 -1 -1 -1 -1 -1  0 -1  0  0 -1 -1 -1 -1 -1  0 -1 -1 -1 -1  0
 0  0 -1  0 -1  0  0 -1 -1  0  0  0 -1  0 -1 -1  0  0 -1  0  0 -1 -1  0
-1 -1 -1 -1 -1 -1  0 -1  1  0  0 -1  1 -1 -1  0  0  0  0  0 -1 -1 -1 -1
 0 -1 -1  0 -1  0 -1  0  0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1  1 -1
-1  0  0 -1 -1 -1 -1 -1 -1 -1  0 -1  0 -1 -1 -1 -1 -1  0 -1 -1 -1 -1
-1 -1 -1 -1  0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1  1 -1 -1  0
-1 -1  0 -1 -1 -1  0  0 -1 -1 -1 -1 -1 -1 -1  0 -1 -1  0 -1 -1  0 -1
-1 -1 -1  0 -1 -1 -1 -1 -1  0 -1 -1 -1 -1 -1 -1 -1  0 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1  0 -1 -1 -1 -1 -1 -1 -1 -1
 0  0 -1 -1 -1 -1  0  0 -1 -1 -1  0  0 -1 -1  0 -1 -1 -1 -1  0 -1 -1
-1 -1 -1 -1 -1 -1  0  0  0 -1 -1 -1  0  0 -1  1 -1 -1 -1 -1  0 -1  0 -1
-1  0  0 -1 -1  0 -1  0 -1  0 -1 -1 -1  0 -1 -1  0  0 -1 -1 -1  0 -1  1
 0 -1  0 -1  0  0  0 -1 -1 -1  1  0 -1 -1 -1 -1 -1 -1 -1  1 -1 -1 -1

```

```

-1 -1 -1 -1 -1 -1 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 0 -1 -1 0 -1 -1 -1
-1 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 0 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 1 -1 -1 -1 -1 0 0 -1
-1 -1 -1 -1 -1 0 0 -1 -1 -1 -1 0 -1 -1 -1 -1 -1 0 -1 -1 0 -1 -1 -1
-1 -1 -1 -1 -1 -1 0 -1 -1 -1 -1 0 0 -1 -1 -1 0 -1 -1 -1 -1 -1 0 1
-1 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 0 0 -1 -1 -1 -1 -1 0 -1 0 -1 -1 -1
0 -1 -1 -1 -1 0 -1 -1 -1 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 0 -1 -1
-1 -1 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 0 -1 -1 -1 -1 -1 -1 0 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 0 -1 0 -1 -1 0 0 0 -1 -1 -1 -1 -1 -1 -1
1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 0 -1]

```

## Valoración Manual

Por ultimo explicaremos la simulación de nuestro sistema de valoración manual. Para ello hemos creado un formulario que permite introducir al usuario una puntuación y ademas un comentario sobre un Pokemon en concreto. Las secciones del formulario son:




- Pokemon: Pokemon a valorar.
- Puntuación: Valor seleccionable de 1 a 5 siendo 5 una valoración muy positiva y 1 una valoración muy negativa.
- Comentario: Comentario adicional sobre el Pokemon valorado.

```

Pokemon = 'Raichu' #@param {type:"string"}
Puntuacion= '5' #@param ["1", "2", "3","4","5"]
Comentario = 'Me parece un gran pokemon con una potencia increihle' #@param {type:"string"}

print(Pokemon)
print(Puntuacion)
print(Comentario)

```

**Pokemon:** " Raichu "   
**Puntuacion:** 5   
**Comentario:** " Me parece un gran p" 

Raichu  
 5  
 Me parece un gran pokemon con una potencia increible

## ▼ BIBLIOGRAFÍA

[Uso de phantomBuster para la generación del csv sobre una busqueda de Twitter](#)

[Información sobre sklearn.metrics](#)

[Información sobre valoración automática en Python](#)

[GitHub con Notebook de Recomendación de películas y Notebook de Analisis de Sentimientos](#)

